Circuit FPGA for Active Rules Selection in a Transition P System Region

Víctor Martínez, Abraham Gutiérrez, and Luis Fernando de Mingo

Abstract. P systems or Membrane Computing are a type of a distributed, massively parallel and non deterministic system based on biological membranes. These systems perform a computation through transition between two consecutive configurations. As it is well known in membrane computing, a configuration consists in a m-tuple of multisets present at any moment in the existing m regions of the system at that moment time. Transitions between two configurations are performed by using evolution rules which are in each region of the system in a non-deterministic maximally parallel manner. This article shows the development of a hardware circuit of selection of active rules in a membrane of a transition P-system. This development has been researched by using the Quartus II tool of Altera Semiconductors. In the first place, the initial specifications are defined in orfer to outline the synthesis of the circuit of active rules selection. Later on the design and synthesis of the circuit will be shown, as well as, the operation tests required to present the obtained results.

1 Theoretical Preliminaries on P-Systems

The Membrane Computing or P Systems (created by Păun [1], [2], [3]) are computation systems based on the biomolecular processes of living cells. According to this, the investigations are based on the idea of the imitation of the procedures that take place in Nature, and their application to machines, can lead to discover and to develop new computation models which will give place to a new generation of intelligent computers. There are many papers about software tools implementing different P-system variants [4], [5] and [6]. However, they are very interesting in order to define hardware implementation of these kinds of systems. Moreover, evolution of transition P- systems is very complicate to be translated into hardware devices due mainly to the membrane dissolving or membrane division capabilities of rules.

Besides that, the non-deterministic maximally parallel manner in which rules are applied inside membranes is more appropriated to be implemented in digital hardware devices. In the case of P-systems hardware implementations only a few papers can be found: a Hardware Circuit for Selecting Active Rules [7], a Cluster of Computers [8], a Master-Slave Distributed Architecture [9] or hardware architecture based on microcontrollers [10].

This article development a FPGA circuit for to select the active rules in a transition P-system membrane, by using the Quartus II tool of Altera Semiconductors [11]. In

the first place, the initial specifications are defined to outline the circuit of the selection of active rules synthesis. Later on, the circuit synthesis will be shown as well as the functioning tests necessary to present the results we have reached.

The definition, according to formal notation, of the membrane system used for the implemented prototype is shown in the following expressions:

$$\Pi = (V, \mu, \omega_1, \ldots \omega_4, (R_1, \rho_1), \ldots, (R_4, \rho_4), 4)$$
$$V = \{a,b,c,d,e\}$$
$$\mu = [_1[_2[_3]_3]_2[_4]_4]_1$$
$$\omega_1 = aac$$
$$R_1 = \{r_1: c \rightarrow (c,in_4),\ r_2: c \rightarrow (b,in_4),\ r_3: a \rightarrow (a,in_2)b,\ r_4: dd \rightarrow (a,in_4)\}$$
$$\rho_1 = \{ r_1 > r_3,\ r_2 > r_3 \}$$
$$\omega_2 = a$$
$$R_2 = \{r_1: a \rightarrow (a,in_3),\ r_2: ac \rightarrow \delta\}$$
$$\rho_2 = 0$$
$$\omega_3 = cd$$
$$R_3 = \{r_1: a \rightarrow \delta\}$$
$$\rho_3 = 0$$
$$\omega_4 = \lambda$$
$$R_4 = \{r_1: c \rightarrow (d,out),\ r_2: b \rightarrow b\}$$
$$\rho_4 = 0$$

The circuit to be carried out obtains the region 1 active rules. The region 1 input values will be the objects multiset w_1, the group of rules R_1, the priority relationships among rules ρ_1 and the inner adjacent regions number (two regions in this case). The objects number V is 5 with decimal multiplicity (0 - 9), therefore, we will need 4 bits to represents each object. And so, a word representing the multiplicity of the 5 objects will occupy 20 bits. The rules group is formed by 4 rules which will be stored in the device memory.

Each of the 4 rules we need to store in the memory will be coded with 64 bits. This is, 20 bits for the antecedent, 20 bits for each consequent of the two inner interior regions, plus 4 remaining bits used to code the priorities mask of each rule with regard to the other ones. The rules are stored, therefore, in a ROM 4x64 bits memory.

2 Hardware Implementation of Active Rules

The logical circuit general description is obtained by means of the conjunction of certain fundamental elements. These elements have been denominated basic Functional Units (*FU's*). These FU's controls the positive validation of a certain evolution rule in function of its applicability, utility and activation properties. The Fig. 1 show the general outline of the circuit based on these functional units.

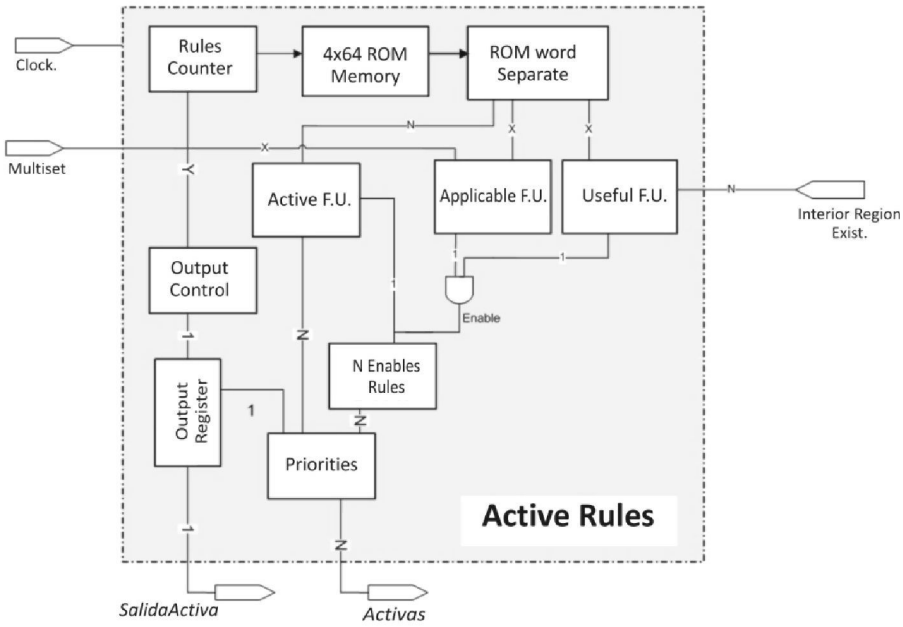

Fig. 1. Scheme general of the circuit to obtain the active rules of a membrane region

Basically, the global circuit inputs are composed by the region objects multiset of the membrane to be evaluated, the in_i bits to determine the interior regions existence, an input clock that allows going on reading and processing the memory rules and a *set* input to initialize the active rules register of the circuit. The register output shows, in positive logic, the active rules of the region, after processing all the ROM memory rules.

In the circuit operation, the memory words will be sequentially read and the different parts that compose each word will be separate. Each different part will be used in the corresponding functional unit or sub-circuit. A counter circuit is needed to read sequentially the memory words. A distributor circuit will be used to separate the different groups of each word.

The usable rule condition is checked by the *Useful Functional Unit* circuit. According to this condition, this particular circuit should check, in each rule, whether the consequent *v* of the rule has some objects which are sent to the inner adjacent regions. In this case, the interior region should exist. The applicability condition of an evolution rule is performed by the *Applicable Functional Unit* circuit. According to the which, the objects of its antecedent should exist in the region multiset in the same or bigger multiplicity.

The *Active Functional Unit* is the part of the circuit in charge of evaluating the priorities among all the existing rules in the membrane region. The result obtained in this unit is a N bits output vector (being N the number of rules). This vector contains the rules potentially active (bit=1) and the rules that should be inhibited in the case of being useful and applicable a rule of more priority that disables it (bit=0).

3 Experimental Results

Next we will present the results of simulating the selection of active rules circuit of the P system example presented in the section 1. For this example, the circuit will obtain the active rules of the region 1, being its two interior regions the 2 and the 4 ones.

On the other hand, the software Quartus II of Inc Altera includes a simulator that can be used to verify as much the behavior as the yield of the carried out designs. The Quartus II simulator allows defining input vectors that will serve as stimulus to the circuit inputs. The application of the input vectors in the simulation allows checking the state of the exits of the circuit, verifying the validity, functionality and effectiveness of the circuit this way.

In Quartus II, we can perform functional simulations, where the circuit output exclusively depends on the inputs values, or we can carry out simulations of a major complexity and depending on time, where one or more clock signals are defined. In our particular case, we are carried out the sequentially reading of the rules defined in ROM memory, with the help of a system clock implemented for that purpose.

To be able to simulate the circuit, the previously described steps for the *Design and Synthesis of the Circuit of Active Rules* must be fulfilled completely:

- Creation of the project "*ActiveRules*"
- Inclusion in the project of all the files with VHDL descriptions of the different elements of the design.
- Inclusion in the project of the schematic blocks files with the design of the circuit.
- Election of the device destination: Cyclone II EP2C35F672C6.
- Circuit Compilation (Synthesis).

As a previous step to the simulation execution, it is necessary to load in the ROM memory system the code of the defined rules for the P system region that is going to be simulated. In this case, the information relative to the rules group R_1, as well as the priorities ρ_1 will be recorded in the ROM memory circuit. The consequents of the regions *here* and *out*, are not necessary for the functionality of the circuit, since they are not evaluated to determine whether a rule is applicable or not.

The information needed to code the region rules, for a certain state, will loaded in words of 64 bits, according to the following distribution:

1. The 20 bits most significant (bits 63 to 44) contain the rule antecedent.
2. The following 20 bits (bits 43 to 24) contain the consequent for the inner region 1.
3. The next 20 bits (bits 23 to 4) contain the consequent for the inner region 2.
4. The last 4 bits of each word of the ROM are used to establish the mask of priorities among the rules defined in the membrane.

In the circuit operation phase, once all the rules have been read sequentially and its evaluation (several clock pulses are needed to address the memory ROM words) has been completed, the *Active Functional Unit* circuit will be responsible for the composing of all the priorities among the rules defined in the system, and for the the

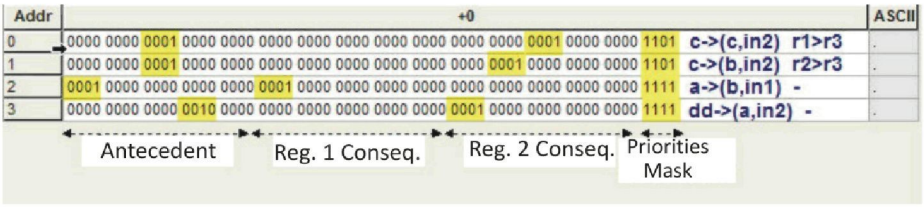

Fig. 2. Shows the method adopted to stored information into the 4x64 ROM memory of the rules R_1 and the priorities masks of the region *1* in the membrane example

\building up of a final mask of priorities, which will be applied to the output vectors proceeding from the other two functional units.

Finally, to check the behavior of the designed circuit, the simulation (*'waveforms'*) vectors are created. They represent the type, the forms and the characteristics of the input signals. In the same way, in the simulation vector the signals representing the output of the circuit we want to evaluate are identified. The simulator applies the test vectors to the compiled design and determines the resulting values in the system outputs.

The next step is to include the circuit inputs and outputs going to be simulated:

- '*Set*': (input, 1 bit). Initialization input of circuit components (ROM, rules counter, Functional Units....)
- *'In1'*: (input, 1 bit). Existence of internal region 1.
- '*In2*': (input, 1 bit). Existence of internal region 2.
- *"w_a", "w_b", "w_c", "w_d", "w_e"*: (input, unsigned decimal, 4 bits). Input Multiset elements.
- '*Clk*': (input, 1 bit). System clock.
- '*Activas*': (output, 4 bits). Exit of the system representing the active rules.
- *'Salida Activa'* (output, 1 bit). Flag determining when the active rules output is stable.

The test case shown corresponds with a computation model in which the circuit of active rules works together with other elements that were changing their conditions according to the temporary evolution of the system. So, we can introduce new aleatory values in the input vectors, by using the *'Set'* flag.

We can check how the *'SalidaActiva'* is validated in irregular periods (due to the *'Set'* activations), and how it becomes stable when there are not new loads of data. The chronogram appeared in figure 3 shows the variation of the inputs signals and the outputs result.

The initial values are:

- w_a [3..0] = 2 (decimal) --> Applicable r_3
- w_c [3..0] = 1 (decimal) --> Applicable r_1 and r_2
- w_d [3..0] = 3 (decimal) > Applicable r_4
- w_b, w_e = 0
- In1 = 1 (Region 1 Exists) --> r_3 is Useful

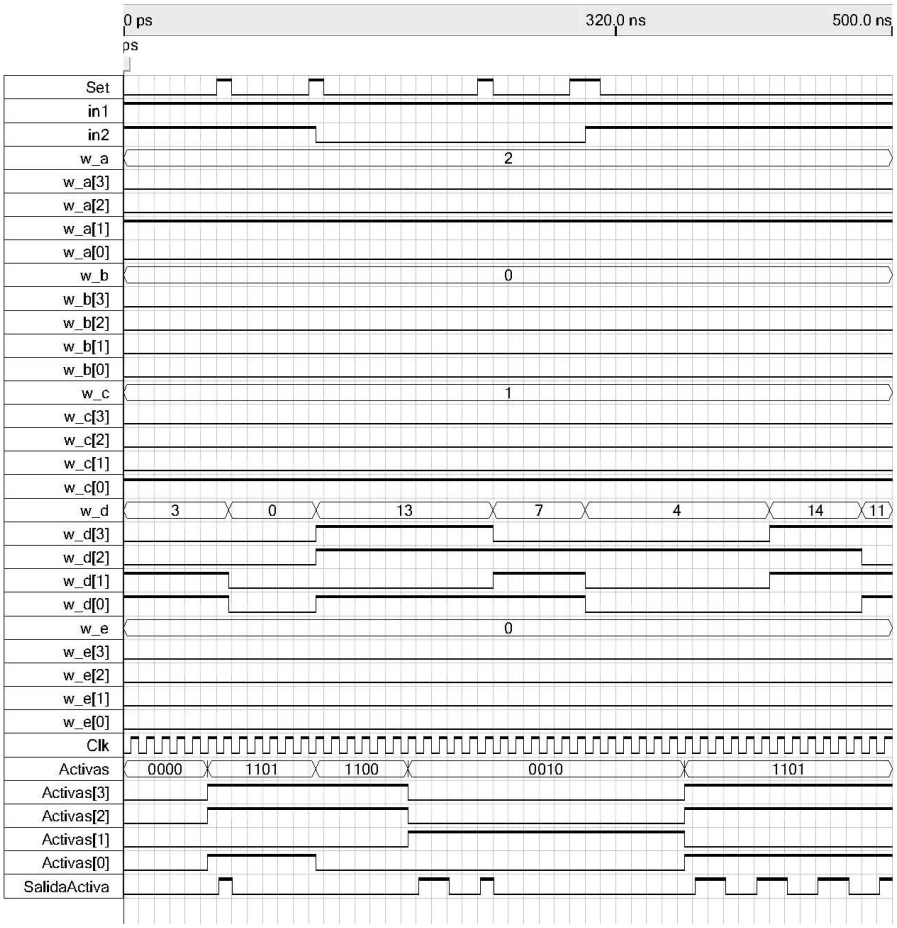

Fig. 3. Inputs-output signals variation

- In2 = 1 (Region 2 Exists) --> r_1, r_2 and r_4 are Useful
- Clock --> Cycle 10ns

Therefore, in accordance with these conditions, the rules r_1, r_2, r_3 and r_4 are all useful and applicable initially. The priorities mask will be 1101, this mask disables r_3. The initial active rules output will have the 1101 value after 5 clock periods showing that the r_1, r_2, and r_4 rules are actives.

An external initialization (*'Set'*) changing the input vectors values to *w_d [3..0] = 0*, would make r_4 not applicable. In this case, the signal *'SalidaActiva'* is annulled by a new change of the *'Set'* signal and the load of new input values. It can be proved how, once the 50ns necessary to complete a new calculation have passed, the change takes place in the vector *'Activas'* and one period later it becomes valid with a 1 in the *'SalidaActiva'* signal. The obtained values remain stable until a new activation of *'Set'* is produced.

4 Conclusions and Future Remarks

This paper presents a direct way to obtain a FPGA circuit capable to select the active rules inside the P-system membrane. The final step is to implement a hardware circuit accomplishing the outlined initial requirement. That is, given an initial multiset of objects, a finite set of evolution rules and an initial Active Rules, the circuit provides the set of rules to be applied to a membrane. The development of the digital system can be carried out by using hardware-software architectures as schematic blocks or VHDL language. The physical implementation can be accomplished on hardware programmable FPGA's devices.

When analyzing the objectives reached, it can be proved that both the designs as much as accomplished synthesis, allow to obtain a hardware system for a general model of P transition system. This means that we can reach the behavior of a region of any membranes system with the obtained circuit. The only limitation is the maximum size of certain parameters, such as the number of rules, the multiplicity of the objects or the number of interior adjacent regions.

The obtained circuit behaves based on the evolution rules stored in memory (which do not change during the system evolution process) and the inputs, which correspond with the values of the region state (reflected in its objects multiset).

If the conditions of the region change, the circuit modifies its outputs being adjusted to the new values. This feature is of a supreme importance in order to integrate this circuit as a module that works cooperatively together with other circuits. These circuits can carry out the rest of the tasks needed to complete the evolution of a transition P system.

References

1. Păun, G.: Computing with Membranes. Journal of Computer and System Sciences 61, 108–143 (2000); Turku Center of Computer Science-TUCS Report No 208 (1998)
2. Păun, G.: Computing with membranes. An introduction. Bulletin of the EATCS 67, 139–152 (1999)
3. Păun, G.: Membrane Computing. Basic Ideas, Results, Applications. In: Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara, Romania, September 26-27, 2005, pp. 1–8 (2005)
4. Arroyo, F., Luengo, C., Baranda, A.V., Mingo, L.F.: A software simulation of transition P systems in Haskell. In: Pre-Proceedings of Second Workshop on Membrane Computing, Curtea de Arges, Romania, August 2002, pp. 29–44 (2002); Păun, G., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) WMC 2002. LNCS, vol. 2597, pp. 19–32. Springer, Heidelberg (2003)
5. Arroyo, F., Luengo, C., Fernandez, L., Mingo, L.F., Castellanos, J.: Simulating membrane systems in digital computers. International Journal Information Theories and Applications 11(1), 29–34 (2004)
6. Fernández, L., Arroyo, F., Castellanos, J., Martinez, V.J.: Software Tools / P Systems Simulators Interoperability. In: Pre-proceedings of the 6th Workshop on Membrane Computing, Vienna, Austria (July 2005)

7. Víctor, J., Martínez, L., Fernández, F., Arroyo, A., Gutiérrez: A Hw Circuit for the Application of Active Rules in a Transition P-System Region. In: Fourth International Conference Information Research and Applications I.TECH 2006, Varna, Bulgaria, Del 20 al 25 de Junio de (2006)
8. Ciobanu, G., Guo, W.: P Systems Running on a Cluster of Computers. In: Martín-Vide, C., Mauri, G., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2003. LNCS, vol. 2933, pp. 123–139. Springer, Heidelberg (2004)
9. Bravo, G., Fernández, L., Arroyo, F., Tejedor, J.A.: Master-Slave Distributed Architecture for Membrane Systems Implementation. In: 8th wseas International Conference on Evolutionary Computing, EC 2007 (2007)
10. Gutiérrez, L., Fernández, F., Arroyo, V., Martínez: Design of a hardware architecture based on microcontrollers for the implementation of membrane system. In: Proceedings on 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006), Timisoara, Rumania, September 2006, pp. 39–42 (2006)
11. Altera Corporation, Silicon Valley, `http://www.altera.com/`